compiler_x@yahoo.com

بسم الله الرحمن الرحيم

الحمد لله رب العالمين

والصلاة والسلام على اشرف الخلق محمد

وعلى ال بيته الحجج المعصومين الطاهرين

اخوتي الكرام :
اضع بين ايديكم هذا الدرس المترجم ليكون ملحق مع الكتيب الذي اطلقته بعنوان الدوال C++ ليكون مدخل الى البرمجة الكائنيه Object Oriented والتي ساحاول قدر الامكان الخوض فيها في كتيب او درس لاحق عسى الله ان ينفعنا واياكم به سائلكم الدعاء الى المولى عز وجل ان يفرج غمة العراق الجريح .

التراكيب structures :

تاتي اهميه التراكيب لتعريف متغيرات بطرق مختلفة عن الطرق المعهودة سابقا فقد نحتاج ان نعبر عن شئ له اكثر من متغير او انه ليس له نوع بياني محدد مثلا لو كنت تعمل في محل واحتجت ان تعبر عن سلعك المعروضه فكل سلعة لها اسم وسعر ورقم تسلسلي و منشا مثلا، فكيف ستعرف السلعة الواحدة هل ستعرفها من النوع الرمزي ام الصحيح ام الحقيقي ام ... فانت تحتاج هنا الى التعامل مع اكثر من نوع بياني للسلعة الواحدة

و التركيب بمفهومه العام هو القيد فهو يعبر عن عدة انواع بيانية مجموعة تحت تعريف واحد .

الصيغة العامة :

```
struct name {
  type1 element1;
  type2 element2;
  type3 element3;
  .
  .
} object_name;
```

Struct : هي كلمة محجوزة للدلالة على انك تعلن عن تركيب.
name : هو الاسم الذي نريد ان نطلقه على التركيب وهو أي اسم خاضع لقواعد التسمية ويمكن في بعض الاحيان اهماله كما سياتي لاحقا .
Type : هو النوع البياني الذي ستعرف فيه عناصر التركيب كان يكون char, int, long او غيرها .
Element : هو العنصر المراد تعريفه وهو يمكن ان يكون أي اسم تضعه شرط ان لا يكون كلمة محجوزة (أي انه المتغير الذي سيكّون عناصر التركيب) ولعدد غير محدد من العناصر وحسب حاجتك .
Object_name : هي معرفات من نوع هذا التركيب(وهي هنا اختيارية) ويمكن تعريفها في هذا الموضع مباشرة او داخل الدالة الرئيسية كانها متغير من نوع تركيب .

يقع التركيب بين جزء ملفات الهيدر headers files وبين جزء الاعلان عن الداله الرئيسية

كيف نعرف متغيرات من نوع تركيب
نستطيع عمل ذلك وطبعا بعد الاعلان عن التركيب وكما قلنا هناك طريقتين لتعريف المتغيرات من نوع هذا التركيب وهما

أ- الاولى داخل جسم الدالة الرئيسية وكأنها متغير عادي مثلا اعلنا عن التركيب التالي

```
# include<iostream.h>

struct products {
  char name [30];
  float price;
} ;



int main ( )
{
      products  radio;
      products  TV, Freezer;
      .
      .
}
```

بـ - اما الطريقة الاخرى لتعريف المتغيرات فتتم مباشرة بعد الاعلان عن التركيب كما ياتي

```
struct products {
  char name [30];
  float price;
}TV,radio ;
```

تلاحظ هنا اننا قمنا بتعريف المتغيرات خارج جسم الدالة الرئيسية بعد جسم التركيب مباشرة بدون الحاجة الى تعريفهم داخل الدالة الرئيسية

ولا فرق بين الطريقتين سوى انك لن تقوم بتعريف المتغيرات من نوع التركيب داخل الدالة الرئيسية كما وانك تستطيع بهذه الطريقة (الثانية) ان تهمل اسم التركيب **products** او تتركه لافرق في الحالتين

```
struct {
  char name [30];
  float price;
}TV,radio ;
```

كما يمكنك تعريف متغيرات بالجمع بين الطريقتين على شرط ان يكون للتركيب اسم :

```
struct products {
  char name [30];
  float price;
}TV,radio ;

int main( )
{
      Products Freezer;
      .
      .
}
```

التعامل مع التركيب :

في الامثله السابقة التركيب اسمه **products** يضم داخل جسمه عنصرين بيانيين هما **name** من النوع **char** و **price** من النوع **float** وهنا نستطيع التعامل مع المتغير كالاتي مثلا اردنا ادخال قيم للمتغير **TV** فيكون بالشكل التالي:

```
cin>>TV.name ;
cin>>TV.price;
```

تلاحظ هنا يذكر المتغير اولا متبوع بـنقطة (.) ثم اسم العنصر المراد استخدامه مثلا **name , price** وتاتي اهمية النقطة (.) هي انه وكماقلنا انه للتركيب عناصر ويذكر بعد هذه النقطة اسم العنصر و الذي يكون هو المتغير الفعلي الذي يتم التعامل معه .

ايضا اود الاشارة الى ان كل متغير مستقل بقيم عناصره فمثلا (**TV.price** ليس له علاقة بـ **radio.price**) فلكل قيمته ويتبع متغير مستقل هو المتغير الرئيسي **TV** او **radio** .

ايضا لو اردت طباعة عنصر المتغير فيتم بالشكل

```
cout<<radio.price ;
cout <<Freeze.name ;
```

ويمكنك اجراء كافة العمليات الحسابية و المنطقية على عناصر التركيب مثلا :

```
Example
Float  total_price;
total_price = TV.price + Freeze.price + radio.price;

Example
if (Freeze.price < TV.price ) { do something};
```

اما الان اعتقد ان الفكرة من التركيب وطرق التعامل معه قد اتضحت وبناء علية سنأخذ مثال متكامل ومختلف قليلا

```
// example about structures
#include <iostream.h>
#include <string.h>


struct movie {
  char title [50];
  int year;
};


int main ()
{
movie mine,yours;
strcpy(mine.title,"Space Odyssey");
mine.year=1968;
cout<<"Enter title\n";
cin>>yours.title;
cout<<"Enter year\n";
cin>>yours.year;

cout << "My favourite movie is:\n ";
cout << mine.title;
cout << " (" << mine.year << ")\n";

cout << "And yours:\n ";
cout << yours.title;
cout << " (" << yours.year << ")\n";
return 0;
}
```

```
Enter title: Alien
Enter year: 1979

My favourite movie is:
 2001 A Space Odyssey (1968)
And yours:
 Alien (1979)
```

المثال يبين كيفية التعامل مع التركيب وعناصره
هنا تم الاعلان عن تركيب اسمة **move** ليعبر عن الافلام (اسم الفيلم **title** وتاريخه **year**) ثم ابتدا البرنامج الرئيسي بتعريف متغيرين من نوع هذا التركيب هما **mine , yours** وتم ادخال القيم الى عناصر المتغير **mine** مباشرة داخل البرنامج (لاحظ ان المتغير الرمزي لانستطيع اسناد القيم اليه بالعلامة = ولذلك استخدمنا الدالة **strcpy(mine.title,"Space Odyssey")** لنسخ محتوى الخيط الرمزي الى المتغير **(mine.title)**
بعد التنفيذ يطلب البرنامج من المستخدم ادخال قيم الى المتغير الثاني **yours** وبعد ان تدخل القيم يقوم البرنامج بطباعة قيم المتغيرين على الشاشة .

كما يمكنك استخدام التراكيب مع الدوال ايضا حيث بامكانك تمرير متغير التركيب كاملا الى الدالة مثلا في المثال اعلاه نستطيع عمل دالة للطباعة اسمها print ونرسل لها متغير التركيب وتقوم بطباعة عناصره

```
// example about structures
#include <iostream.h>
#include <string.h>

struct movie {
  char title [50];
  int year;
} mine, yours;

void print ( movie mymovie);

int main ()
{
 movie mine;
 strcpy(mine.title,"Space
Odyssey");
 mine.year=1968;
 cout<<"my favorite movie is :\n";
 print(mine);
 return 0;
}

//  define function
void print (movie mymovie)
{
 cout << mymovie.title;
 cout << " (" << mymovie.year <<
")\n";
}
```

```
my favorite movie is :
Space Odyssey (1968)
```

كما يمكنك ايضا تمرير عنصر واحد او اكثر الى دالة على ان يعرف باراميتر الدالة من نفس نوع العنصر داخل التركيب وليس من نوع التركيب
مثلا لو اردنا تمرير العنصر title فقط الى الدالة لطباعته فسيكون البرنامج بالصورة التاليه

```
// example about structures
#include <iostream.h>
#include <string.h>



struct movie {
  char title [50];
  int year;
} mine, yours;

void print_title ( char
s_title[50]);

int main ()
{
 movie mine;
 strcpy(mine.title,"Space
Odyssey");
 cout<<"my favorite movie is :\n";
 print_title(mine.title);
 return 0;
}

void print_title (char
s_title[50])
{
 cout <<s_title<<endl;
}
```

```
My favourite movie is:
Space Odyssey
```

احدى خصائص التراكيب هو انه يمكن استخدام جميع عناصرها او جزء منها ففي المثال اعلاه استخدمنا فقط العنصر **title** واهملنا **year** وذلك حسب حاجتنا في البرنامج .

في هذا المثال سننشئ قاعدة بيانات بالاعتماد على التراكيب
قمنا سابقا بتعريف متغيرات من نوع تراكيب مثلا التركيب التالي يعرف طالب على اساس معلوماته

```
struct info{
char name [50];
int age;
long no;
};
```

والان لو اردنا ان نعرف طالب من هذا النوع فسنعرفه كالتالي

```
info student;
```

ليست هناك أي مشكلة ولكن ، لو اردنا تعريف عشرة طلاب فهل سنكتب عشر متغيرات كما في الشكل

info student1, student2, student3, student4, ……, student10

وتخيل لو انك اردت ادخال العناصر للطالب الاول فسيتحتم عليك ان تكتب

cin>>student1.no>>student1.name>>student1.no;

ولو اردت ادخال البيانات لجميع الطلاب فسوف يتحتم عليك كتابتها عشر مرات وكذلك لو اردت طباعتهم ناهيك عن عمليات المعالجة و المقارنات و التعديل و و و و
فهل هذه طريقة عملية في البرمجة وهل تدل على مهارة او احتراف ؟ بالتاكيد لا
اذن لا بد من وجود طريقة اخرى ابسط واقصر وهي باستعمال المصفوفة ولا شئ سيختلف سوى انك لن تعرف متغير من نوع info ولكنك ستعرف مصفوفة من هذا النوع info وكما يلي

info student[10];

اصبحت لدينا مصفوفة من نوع تركيب info ولادخال بيانات الى العنصر الاول من المصفوفة

cin>>student[0].name;
cout<<student[0].age;

وهكذا بقية العناصر والذي سيختلف فقط دليل المصفوفه **0,1,2,3,....**
ولطباعة العمر مثلا

cout<<student[0].age;

وهكذا بقية العناصر

وكما تعرف انك تستطيع استخدام العدادات مع المصفوفة ولذلك تستطيع التعامل مع كل عناصر المصفوفة بسهولة ويسر

شاهد المثال التالي

```
// example about structures
#include <iostream.h>
#include <string.h>

#include <iostream.h>

struct info{
char name [50];
int age;
long no;
};

int main ()
{
 info student[10];
 for(int i=0;i<10 ;i++)
  cin>>student[i].no>> student [i].name>> student [i].age;

 for( i=0;i<10 ;i++)
{
cout<< student [i].no<<" "<< student [i].name<<" "<<
student [i].age <<endl;
cout<<"-----------------\n";
}

return 0;
}
```

وبذلك تستطيع انشاء قاعدة بيانات كاملة بهذه الطريقة كما يمكنك تعديل البرنامج لاضافة ميزات اخرى اليه مثل البحث عن اسم طالب او معرفة الطلاب ذوي اعمار محددة و الكثير

التراكيب المتداخلة

يمكن ان يكون هناك تركيب يحتوي على تركيب اخر (أي انه يكون هناك تركيب يحتوي على عنصر هذا العنصر من نوع تركيب ايضا) وهذه تسمى التراكيب المتداخلة
مثلا

```
#include<iostream.h>

struct s_name{
 char first[20];
 char father[20];
 char family[20];
};

struct s_student{
 s_name name;
 int av;
};

int main()
{
 s_student stud[3];

for (int i=0;i<3;i++)
{
 cout<<"Enter student name ";
 cin>>stud[i].name.first;
 cout<<"Enter family name ";
 cin>>stud[i].name.family;
 cout <<"Enter av ";
 cin>>stud[i].av;
}
cout<<"------------------------\n";
for (i=0;i<3;i++ )
 {
  cout<<stud[i].name.first;
  cout<<" "<<stud[i].name.family;
  cout<<" "<<stud[i].av<<endl;
 }
return 0;
}
```

المثال ابتدا بالاعلان عن تركيب اسمه s_name تتكون عناصرة من الاسم الاول first و اسم الاب father و اسم العائلة family ، ثم تم تعريف تركيب اخر اسمة s_student تم فيه تعريف عنصرين الاول هو الاسم name من نوع التركيب s_name المعلن عنه وبذلك يمكن ان ياخذ العنصر name احد القيم التالية

name.first
name.father
name.family

وحسب حاجتنا في البرنامج (مثلا في مثالنا استخدمنا الاسم الاول واسم العائلة فقط)
بعدها تم الاعلان عن تركيب اسمه s_student والذي يحتوي على عنصرين العنصر الاول name من نوع التركيب s_name والذي من الممكن ان ياخذ ثلاث عناصر كما ذكرنا ، و العنصر الثاني av من النوع الصحيح ، اما في الدالة الرئيسية فقد ابتدئنا بالاعلان عن مصفوفه اسمها stud من نوع التركيب s_student وتم استخدام عداد (counter) لادخال القيم اليها وكما تلاحظ .

حيث للتعامل مع العنصر name من المعرف [stud[i ينبغي وضع نقطة (.) بعد [stud[i وبعدها كلمة name ولما كان name من نوع تركيب ايضا فانه يتبع بنقطة اخرى لاننا اصبحنا الان داخل التركيب s_name ثم بعد النقطة نكتب العنصر المراد وهو في مثالنا first او family

cin>>stud[i].name.first

اما العنصر av التابع للتركيب s_student فانه لادخال القيم اليه يذكر العنصر المعرف مثلا [stud[1 متبوع بنقطة ثم av مثلا:

stud[i].av;

ويمكن توضيحه بمخطط كالتالي :

```
Struct s_student
{
  S_name name;  ------>  .First
  int av;                .Father
}                        .Family
```

اعتقد انه يمكننا القول الان ان هذا الموضوع انتهى ونحن مستعدين الان بهذا الموضوع وموضوع الدوال للدخول الى البرمجة الكائنية المنحى oop .

تم بحمد الله

مثنى عبد الرسول محسن الفرطوسي
كلية شط العرب الجامعة
٢٠٠٦
Compiler_x@yahoo.com